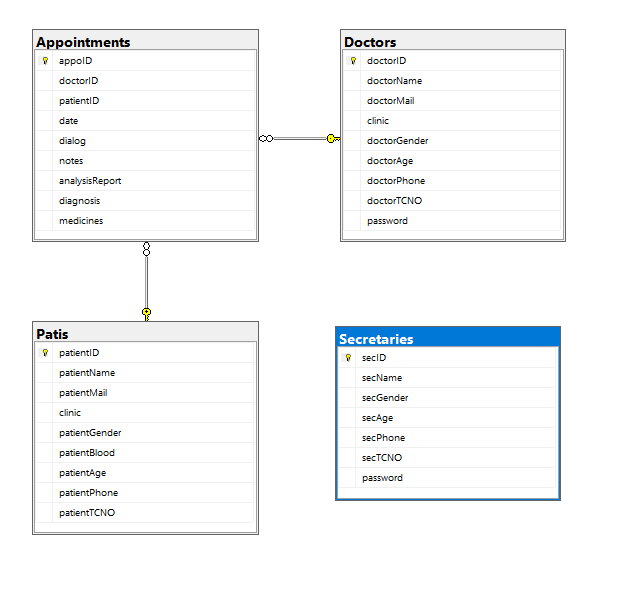
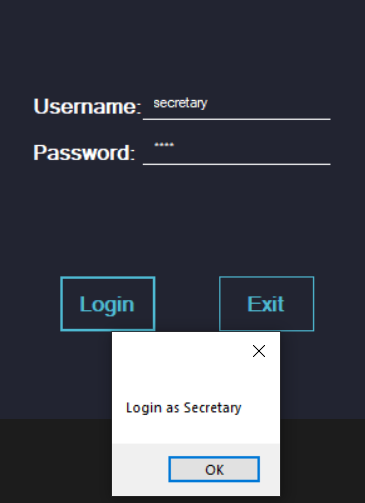
**DÜZCE ÜNİVERSİTESİ**

**GÖRSEL PROGRAMLAMA PROJE RAPORU**

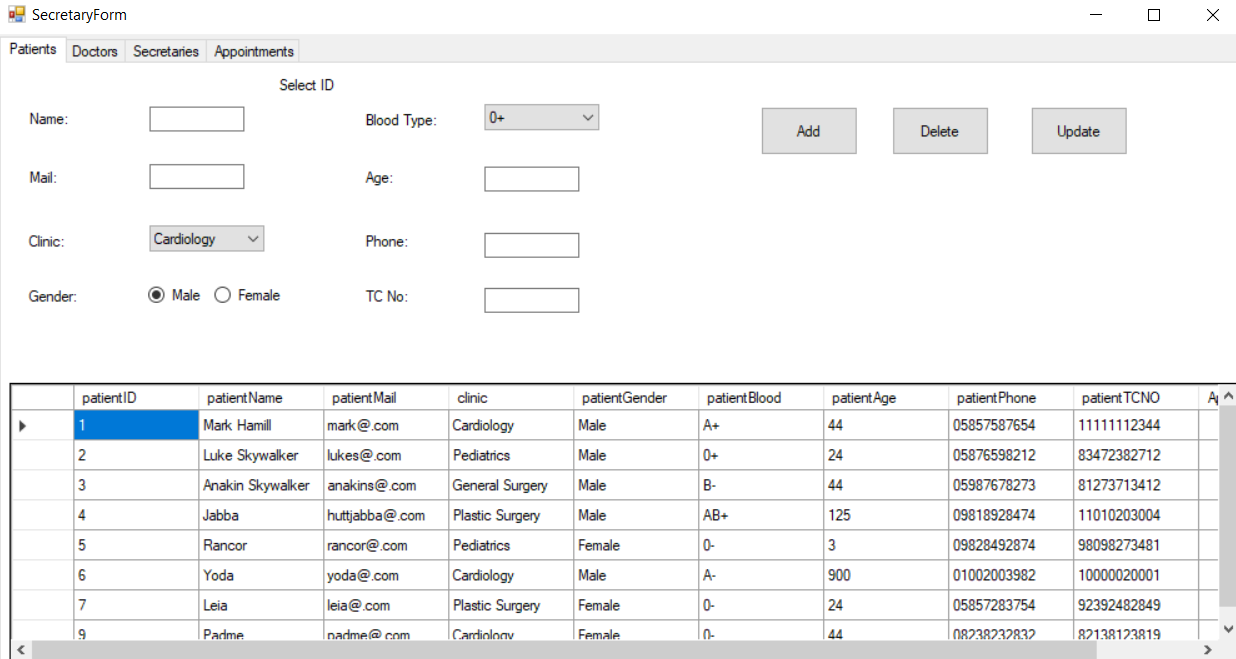
Visual Studio 2019 ve Microsoft SQL Server Management Studio 18 kullanılarak hazırlanmıştır.



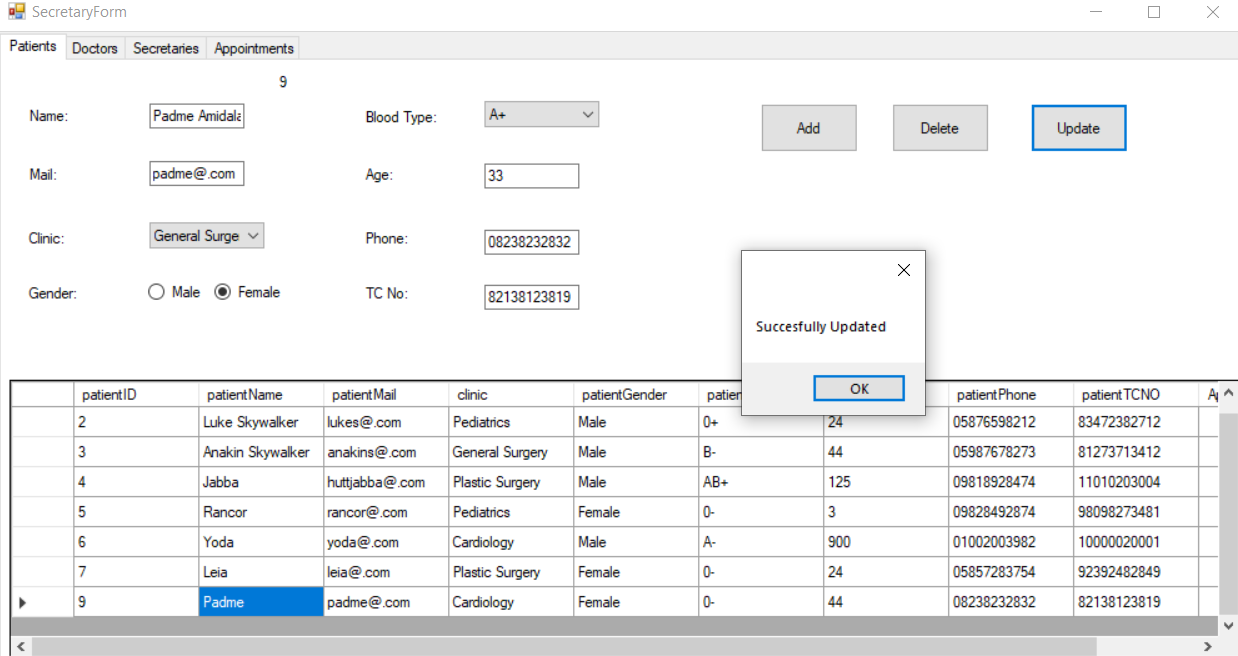
Veritabanı diyagramım yukarıda görüldüğü şekildedir. Appointment tablosundaki doctorID ve patientID Doctors ve Patis tablolarında foreign key olarak alınmıştır. 4 tablonunda primary keyleri ID lerdir ve 1’er 1’er otomatik artarlar.



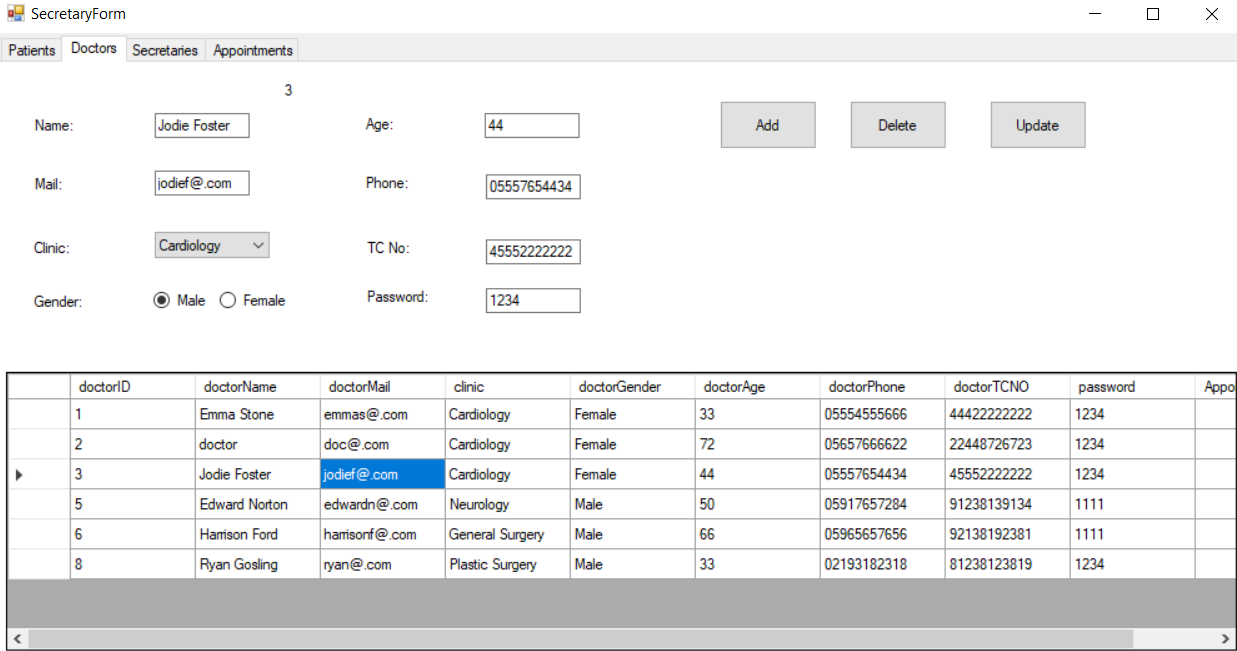
Login ekranında Secretaries ya da Doctors tablosundaki kaydın name ve password verilerini kullanarak giriş yapıyoruz. Yanlış giriş yaparsak hata alıyoruz. Eğer Secretaries tablosundaki veri ile giriş yaparsak SecretaryForm, Doctors tablosundaki veri ile giriş yaparsak DoctorForm açılıyor. Ekrandaki “secretary” username i benim sadece giriş kısmı kolay olsun diye eklediğim test için kullanılıyor (Aynı şekilde Doctor Tablosunda “doctor” isimli kayıtta da var). Normal girişler de username gerçek isimleri baz alır. Ayrıca doktor olarak giriş yaparsak, giriş yapan doktorun id si de alınıyor. Bunun sebebi doktora kendisinden randevu alan hastalarla ilgili işlem yapabilme yetkisi sunmaktır.

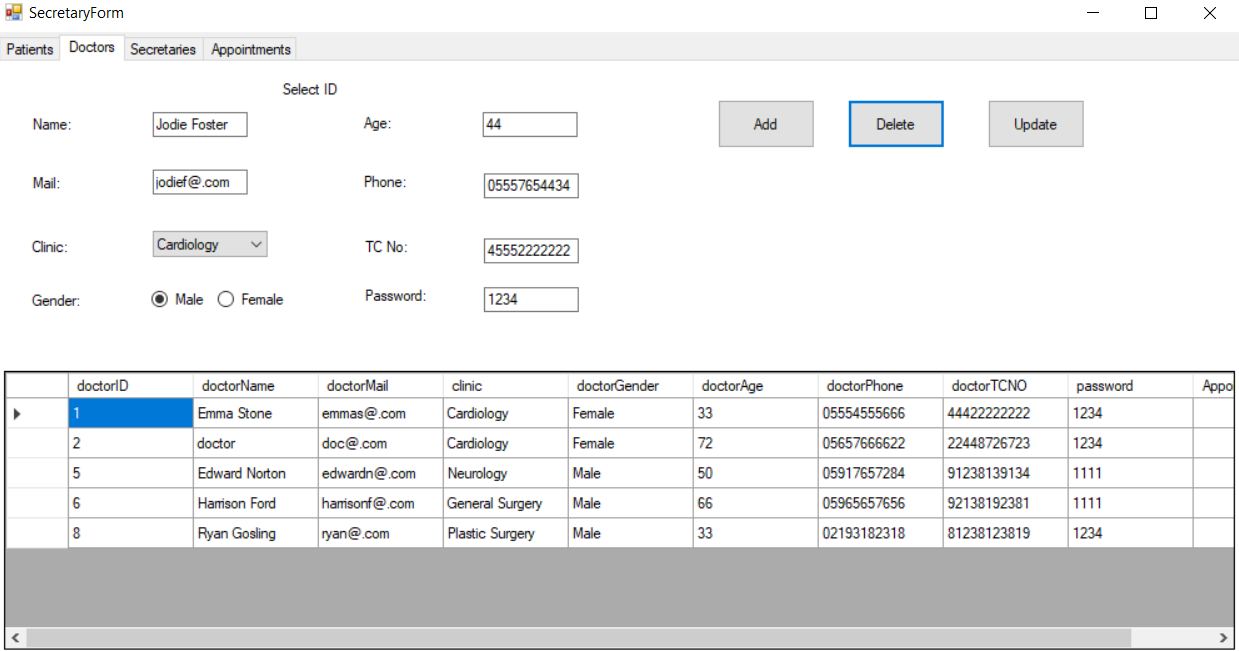


Sekreter olarak giriş yaptığımızda ekrana çıkan ekran yukarıdaki şekildeki gibidir. Bu ekranda Sekreterler, hasta, doktor, sekreter ve randevu oluşturup, güncelleyip, silebiliyorlar. Silme ve güncelleme yapmak için öncelikle aşağıdaki tablodan işlem yapmak istediğimiz kaydı seçiyoruz. Bir kayıta tıklayınca o kayıtın tüm bilgileri giriş yaptığımz textboxlar, comboboxlar ve radiobuttonlara da geliyor.

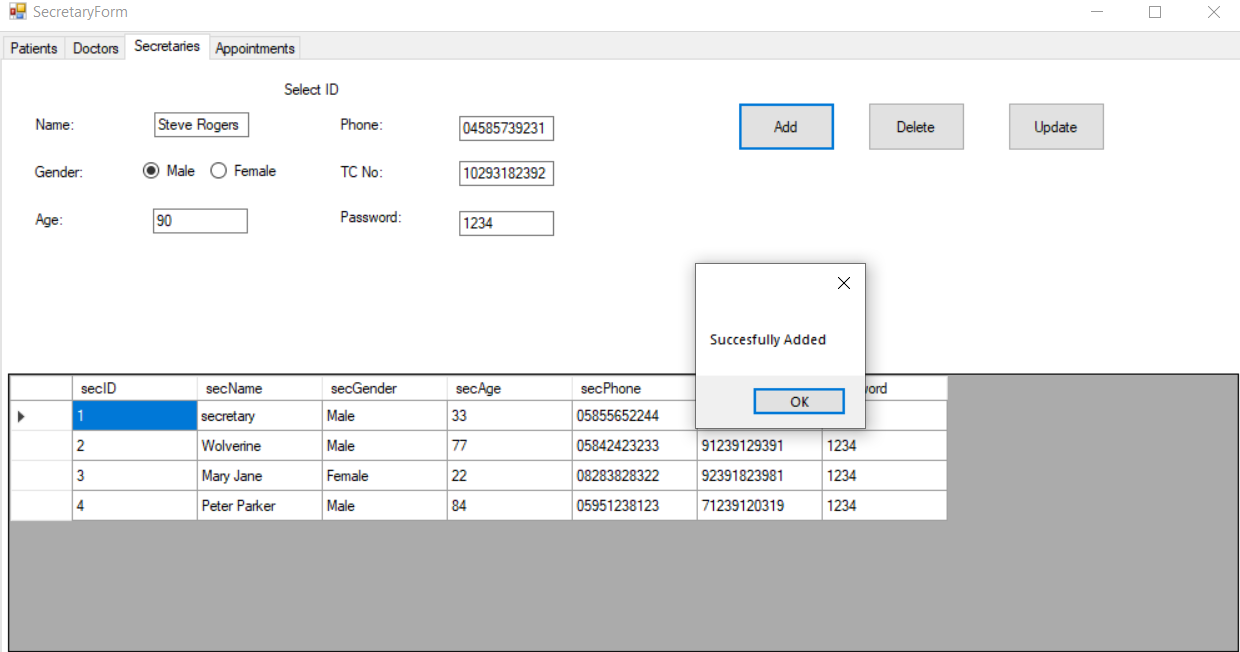


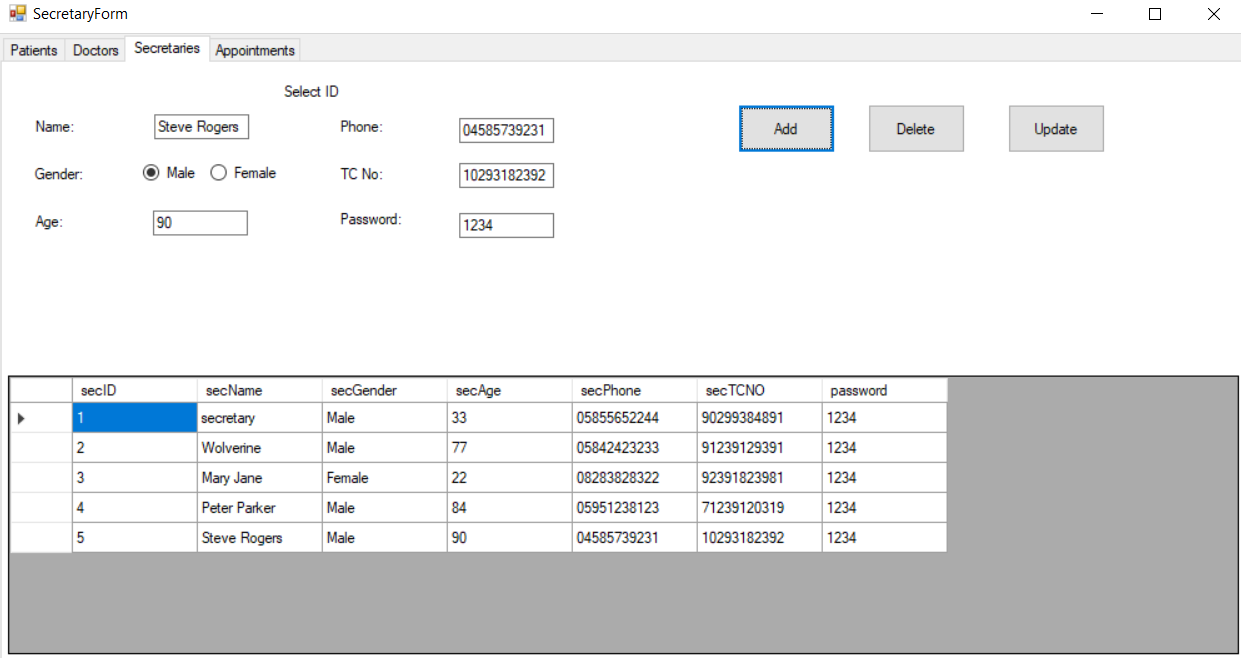
Yukarıdaki şekilde önce padme kaydını seçtim ve padme ile alakalı tüm bilgiler ilişkili yerlerine geldi. Böylece ben sadece değiştirmek istediğim Age, Name, Clinic ve Blood Type seçeneklerini değiştirip Update e bastım. OK tuşuna bastıktan sonra aşağıdaki tablo da bilgileri tekrar çekerek güncelleniyor.



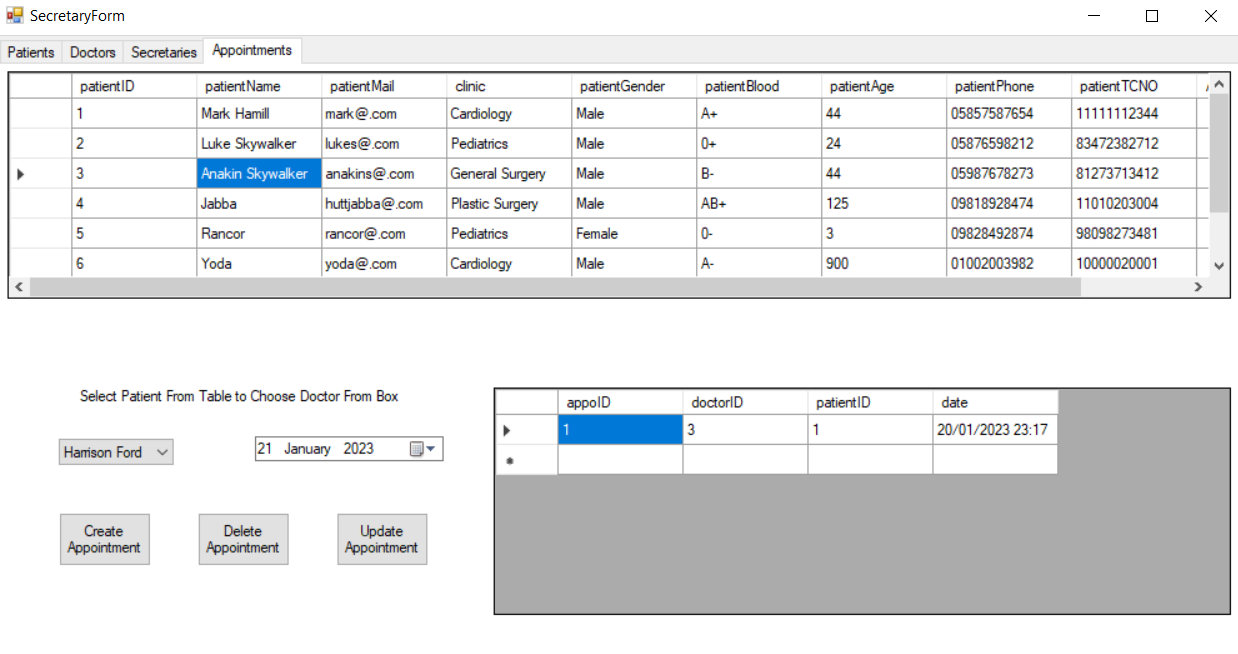


Yukarıdaki Şekillerde önce 3 id li doctoru seçip delete bastığımda silindiğini görüyorsunuz. Eğer kayıt seçilmemişse Hata verip silmeden önce kayıt seçmeniz gerektiğini söyleyen mesaj gösterir.



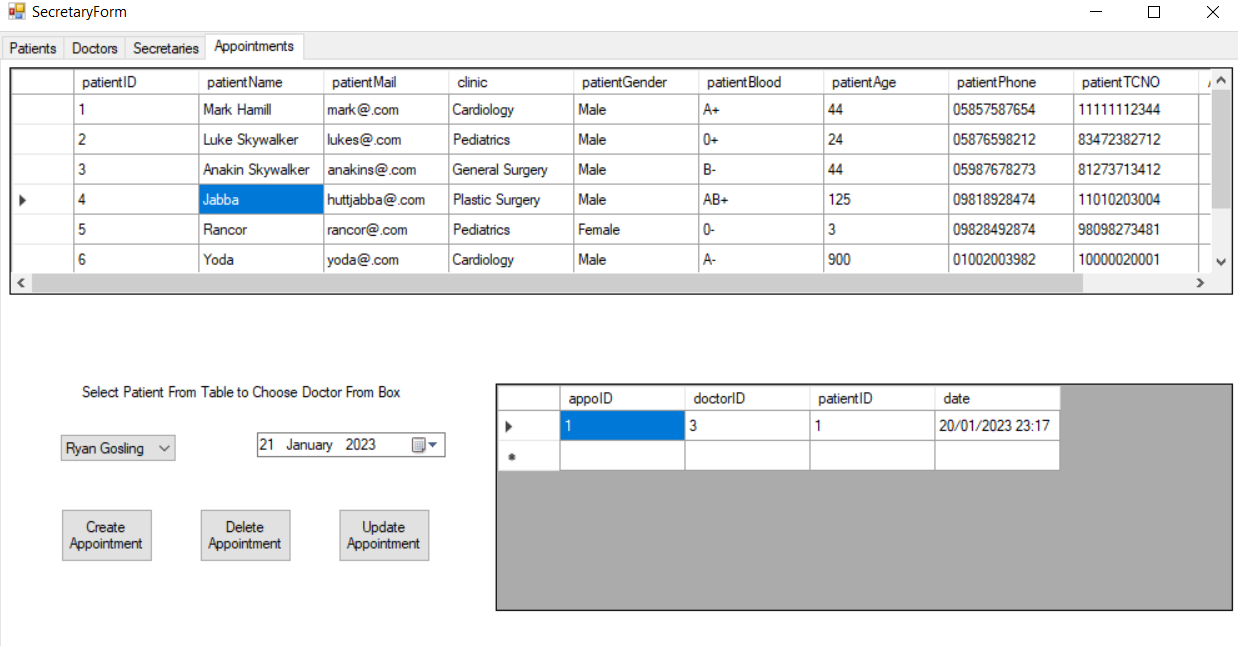


Yukarıdaki şekillerde ise bilgileri girip Add e bastıktan sonra Kaydın eklenip tablonun güncellendiğini görüyorsunuz. Eğer bilgiyi yanlış girseydiniz hata alırdınız. Bilgileri denetleme kısmını kodları açıkladığım kısımda detaylı anlatacağım.

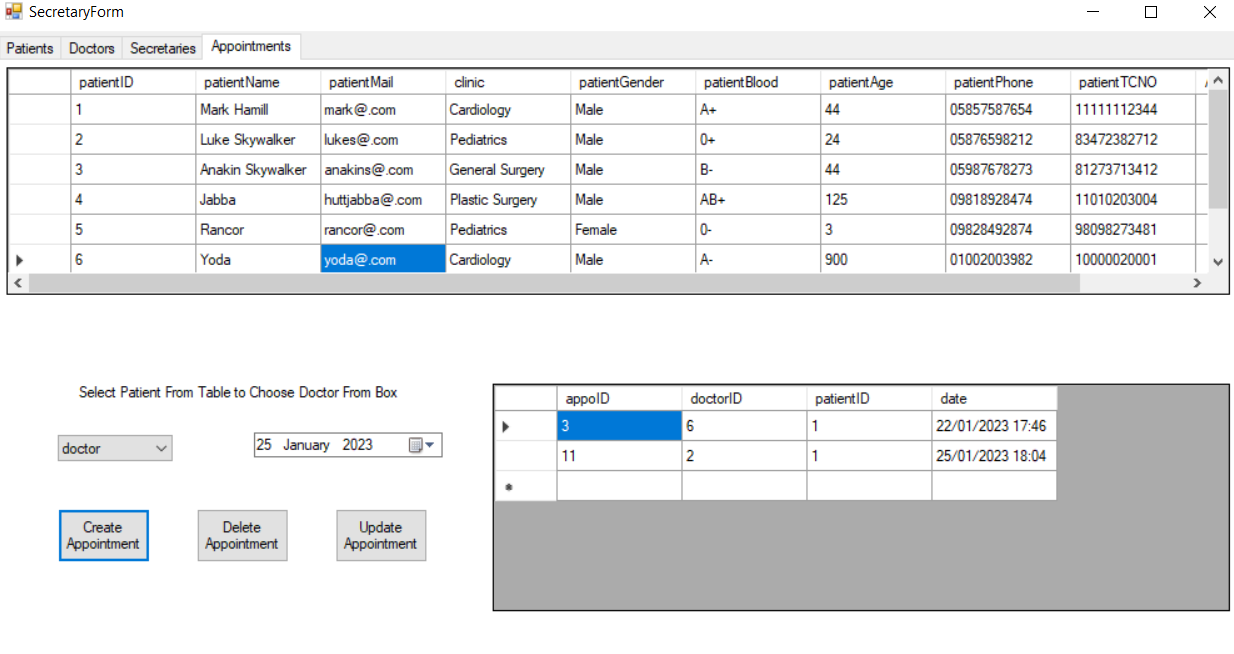


Yukarıdaki şekilde Randevu sekmesini görüyorsunuz. Bu sekmede sekreterler, hastaları ilgili doktorlar ile eşleyip randevu oluşturabiliyorlar. Yukarıda duran tabloda hastalar gözükmektedir. Şuan Anakin Skywalker kaydı seçilidir ve kaydın hastalığı “General Surgery” kliniğindedir. Bu yüzden comboboxa General Surgery alanında çalışan tüm doktorlar eklenir. Eğer ilgili bölümde doktor yoksa program seçtiğiniz bölümde kayıtlı doktor yok diye mesaj gösterir.

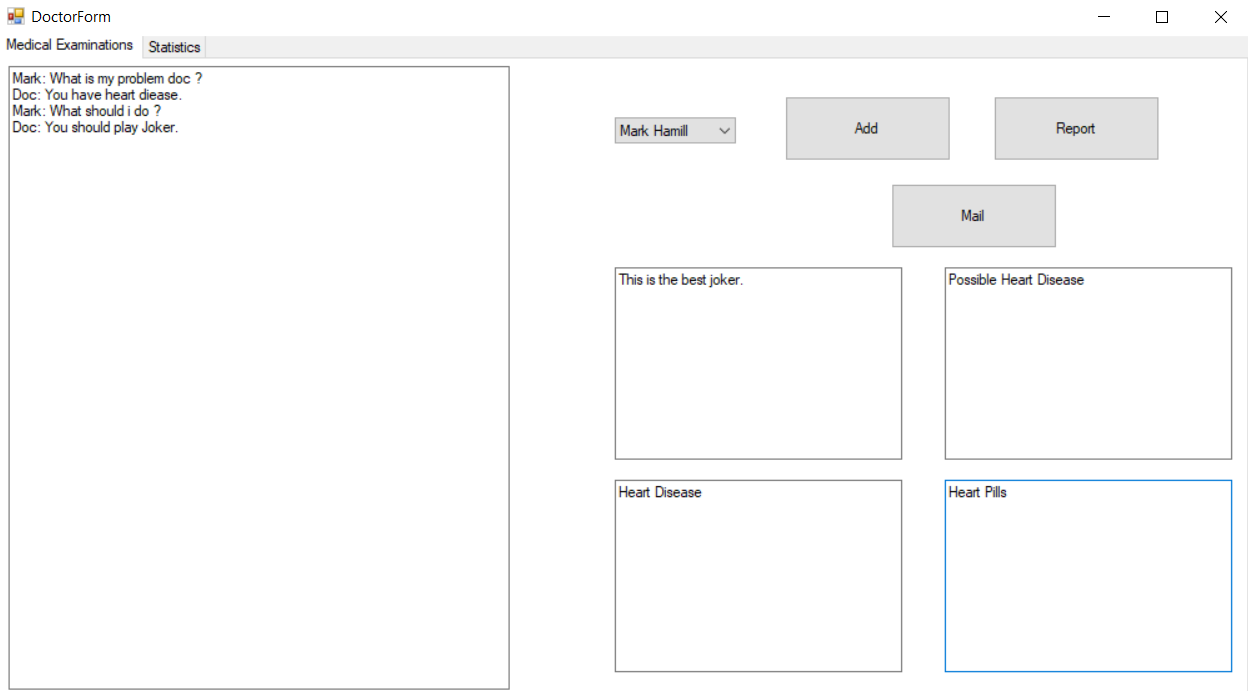
Eğer aşağıdaki şekildeki gibi başka bir kliniğe gitmesi gereken hasta seçilirse comboboxtaki doktorlar da bu uzmanlığı bu klinikte olan doktorlar olarak değişir.



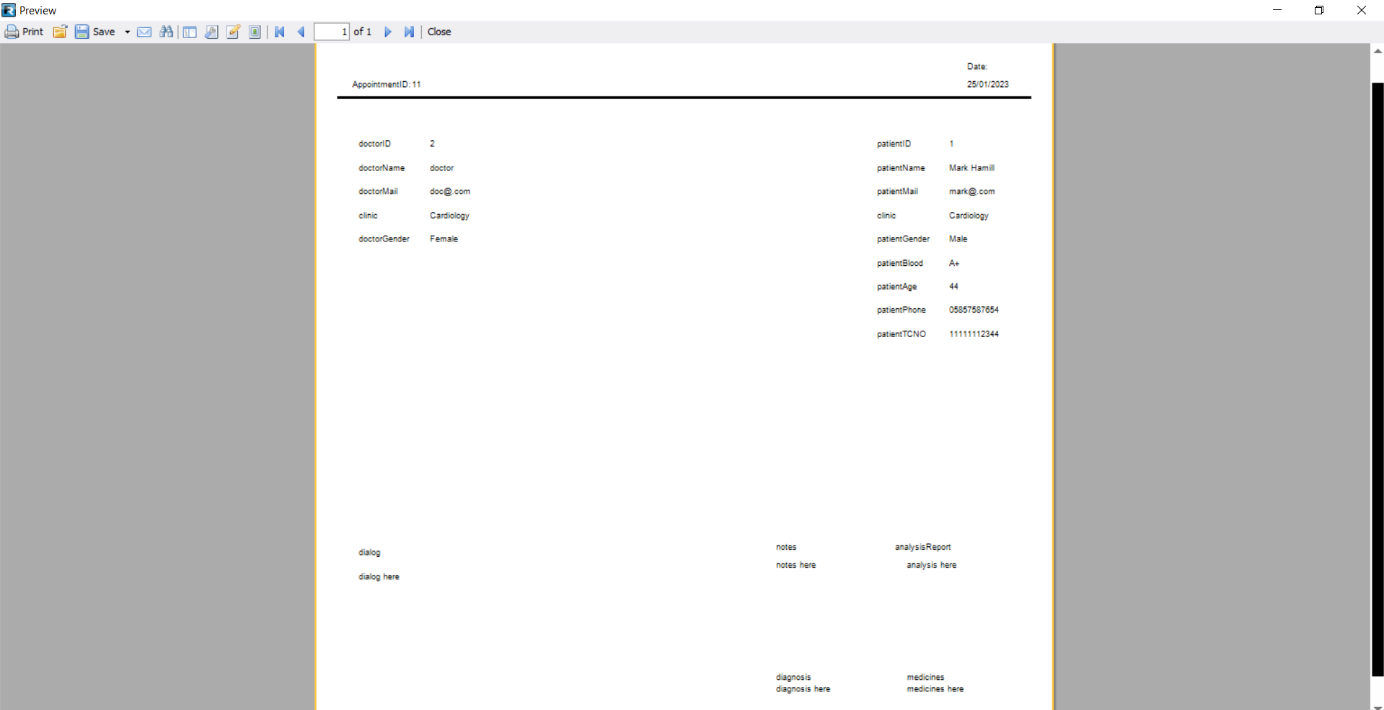
Tablodan hasta ve combobox tan da doktoru seçip tarihi seçtikten sonra create appointment a tıklanarak appointment oluşur.



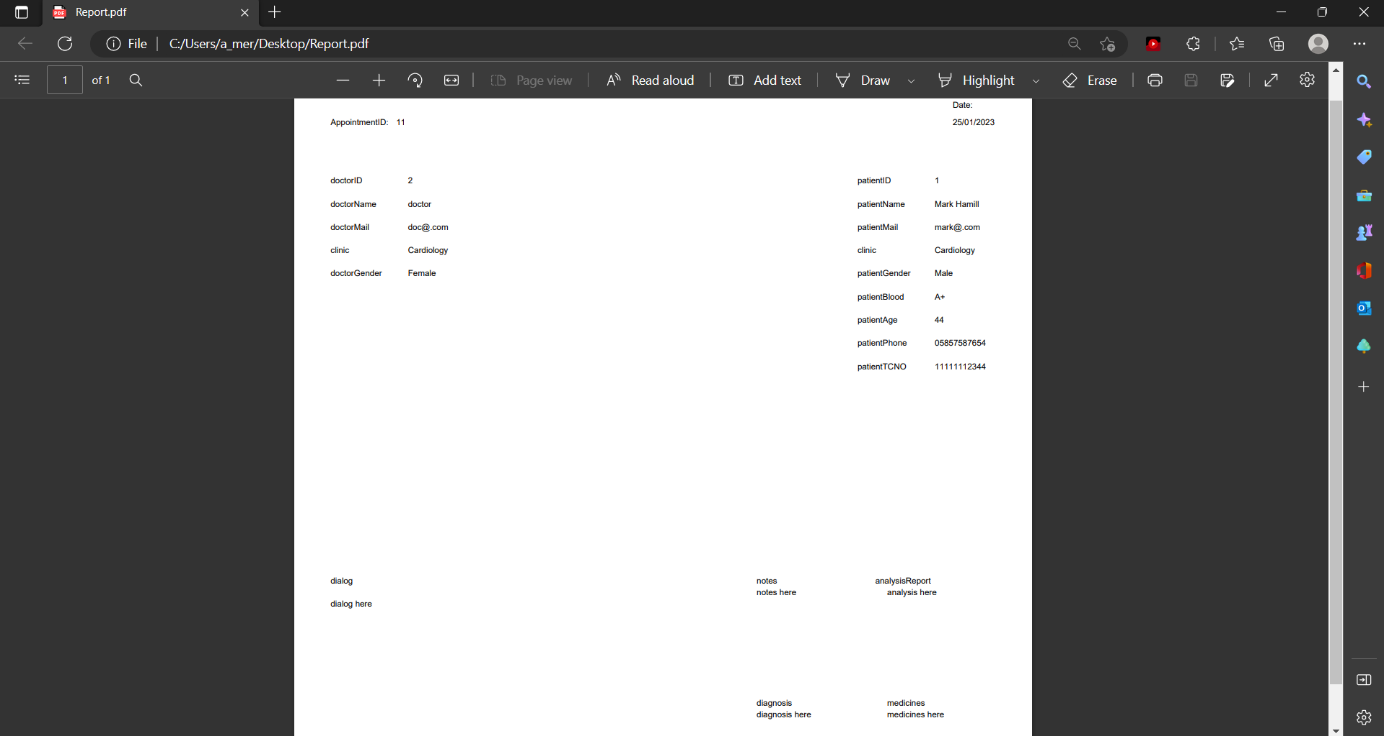
Yukaridaki şekilde yoda ile doctor isimli doktorun kaydını oluşturduğumu görüyorsunuz. Delete ve Update tuşları da diğer tablolarda çalıştığı gibi çalışmaktadır.



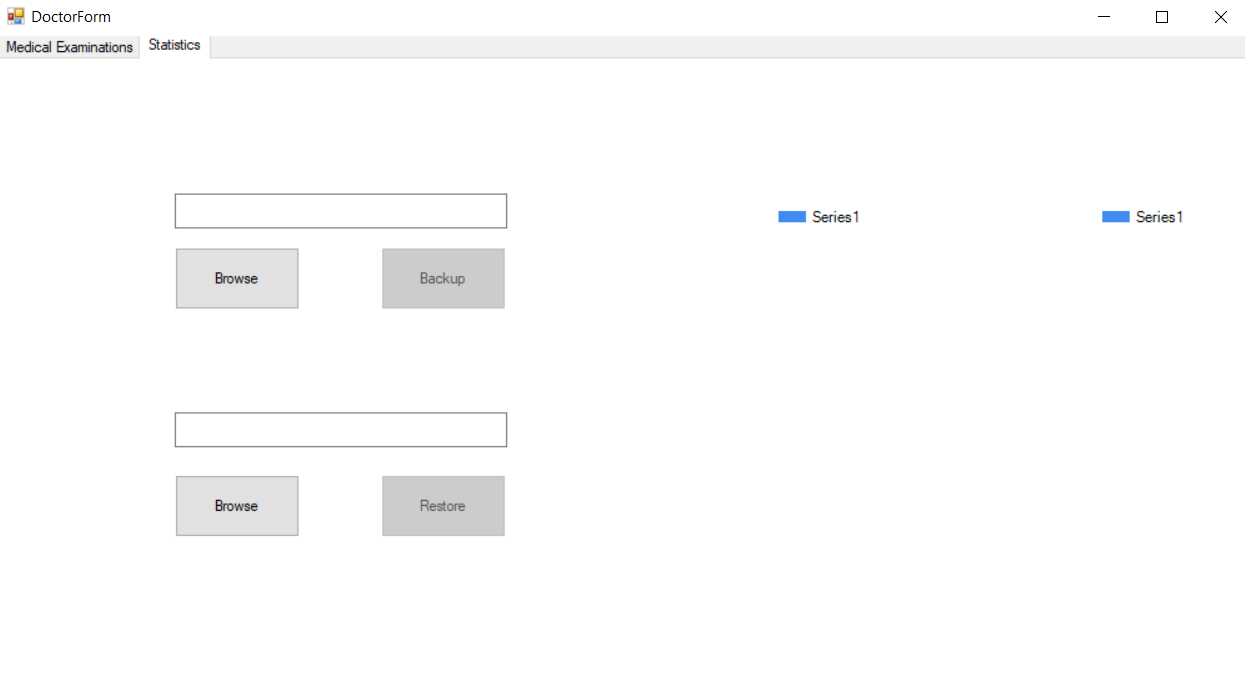
Sisteme doktor olarak girince de şekilde ekranı görüyoruz. Bu ekranda eğer doktorun kayıtlı randevusu varsa comboboxa ekleniyorlar ve doktor oradan hastayı seçip konuşma kaydı, notlar, ilaçlar, hastalık teşhisi, tahlil sonuçlarını girebiliyorlar.



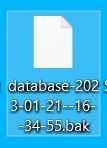
Daha sonra rapor tuşuna basarak yukarıda oluşturduğum şekilde bir rapora erişebiliyorsunuz. Bu raporu FastReport olarak oluşturdum ve buradan raporu pdfe, doc vb pek çok uzantılı dosyaya çevirebilir, yazdırabilir ve mail olarak gönderebilirsiniz.



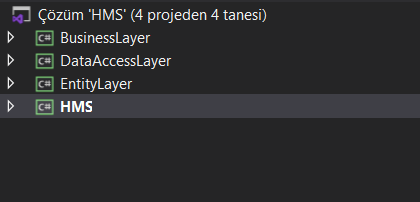
Yukarıdaki şekil doktor ile hastanın arasındaki diyalogu ve bilgilerini de barındıran dosyanın pdf olarak kaydedilmiş hali.



Yukarıdaki şekil ise son ekrandır. Burada normalde grafikler bulunması gerek ancak bunlar çalışmıyor. Sol taraftakiler ise tüm veritabanını backup-restore etmeye yarayan kısımdır. Önce browse ile dosyayı backup edeceğimiz yeri seçip backup ediyoruz. Daha sonra da restore ile bu dosyayı seçip restore tuşuna basınca veritabanını restore ediyoruz. Bu şekilde kaydedilmiş bir backup dosyası aşağı şekilde gösterilmiştir.



**Kodların Açıklanması:**

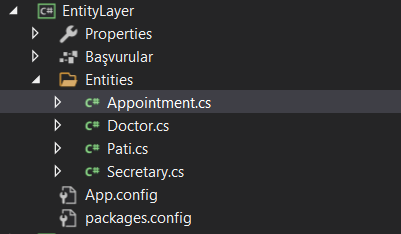


Yukarıdaki şekilde görüldüğü üzere proje 4 katman oluşuyor.

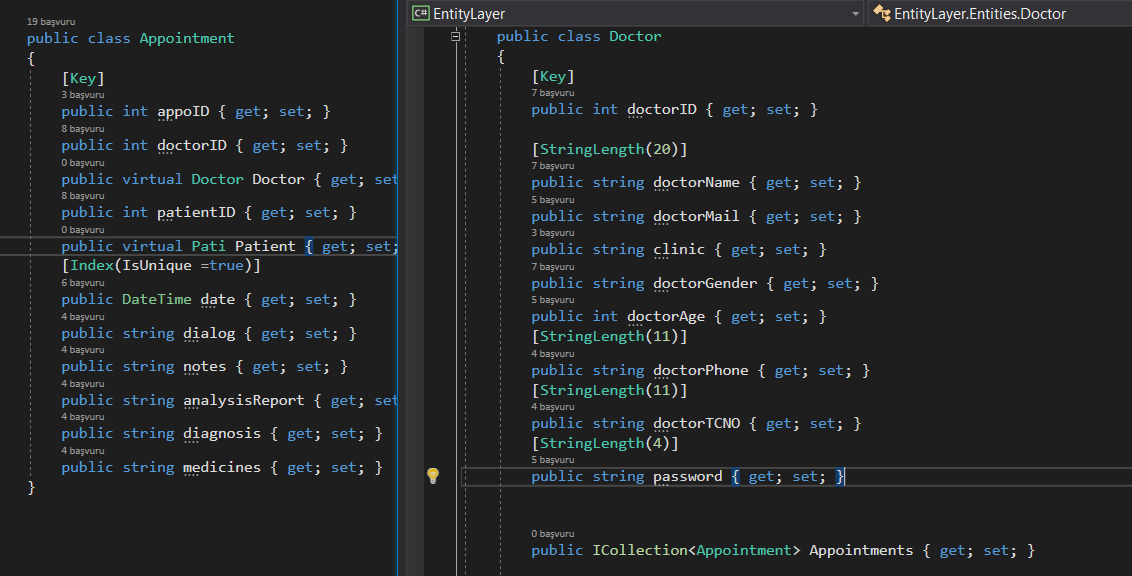
HMS (Hopital Management System): Formların görüntülendiği kullanıcının göreceği kısımdır (Frontend).

**EntityLayer:**

Aşağıdaki şekilde de görüldüğü üzere tablolarımın oluşturulduğu kısımdır. Bu kısımda tablolar, içlerindeki değişkenler, primary keyler, sınırlamalar ( [StringLength(10)] vb. ) ve tablolar arası ilişkiler bulunmaktadır.

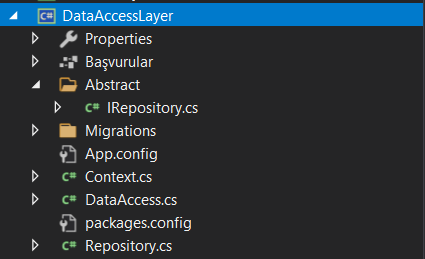


Aşağıdaki şekilde ise Appointment ve Doctor tablolarımı görüyorsunuz. Burada Appointment tablosu Doctor ve Patient tablolarından idleri foreign key olarak alıyor. Böylece bu tablolarla ilişki kurmuş oluyor.

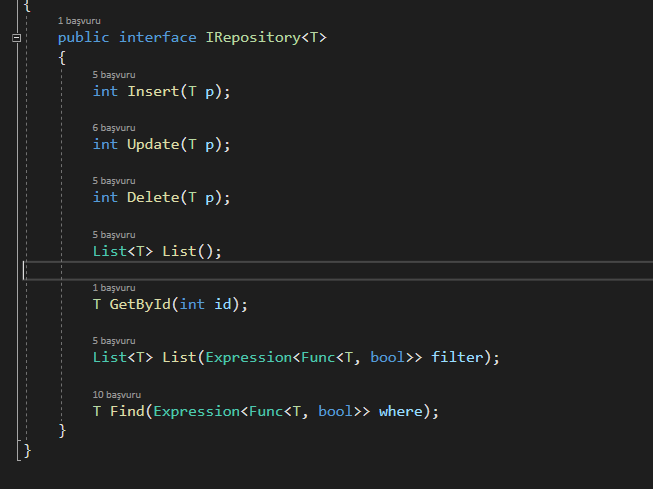


**DataAccessLayer:**

Bu katman ise veri erişim katmanıdır. Burada tablolar üzerinde yapılabilecek işlemler tanınır.



Aşağıdaki kısım IRepository interface’idir. Burada tanımlanan işlemler tablolar üzerinde yapılabilir hala getirilir. Mesela List işlemi List döndüren bir fonksiyon gibidir. Bunu Entity parametresi ile çağırdığımızda o tablodaki tüm verileri döndürür.



Aşağıda ise Context sınıfını görüyorsunuz. Burası tabloların asıl oluştuğu yerdir. Context kısmı DbContext ten miras alarak entitylerimizi tablolar dönüştürür. Basit bir değişken ataması ile karşılaştıracak olursak:

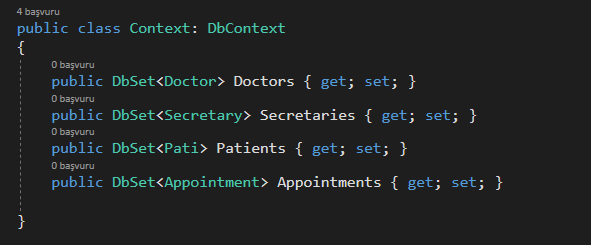
int x; için

DbSet<Doctor> = int kısmı

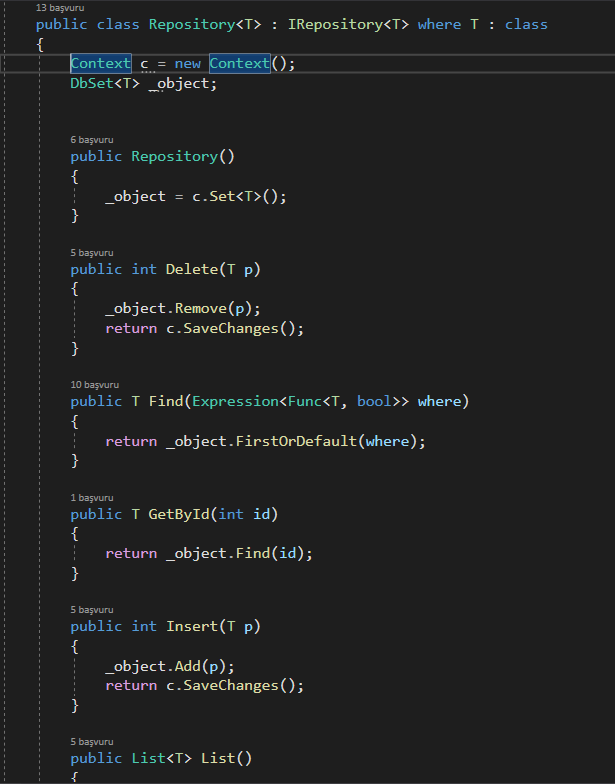
Doctors = x kısmıdır

Yani Entity türünde Doctors adlı bir değişken atadığımız söylenebilir.

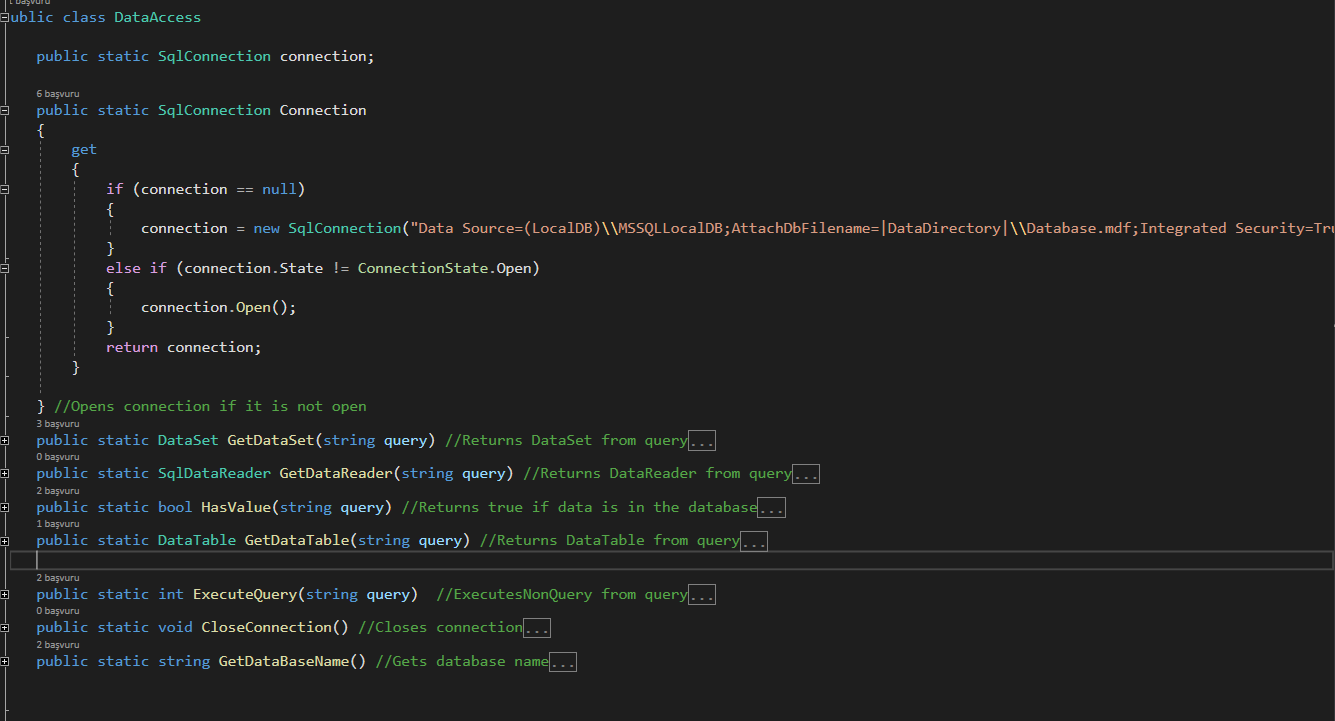
Bu işlemi yapmamızı sağlayan ise DbContext ten miras aldığımız DbSet sınıfıdır.



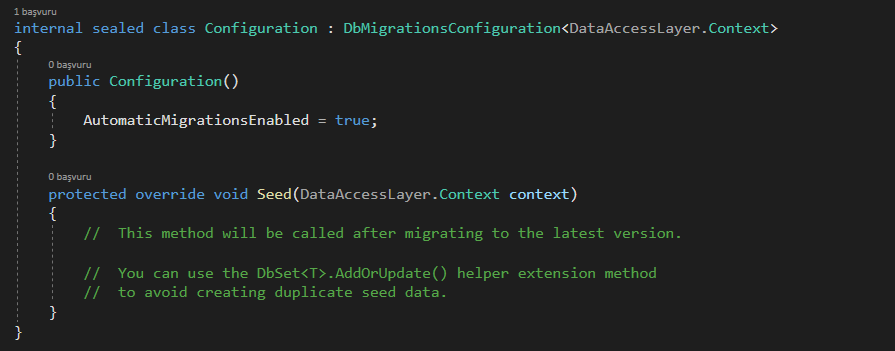
Aşağıdaki şekilde ise Repository sınıfını görüyorsunuz. Burası IRepository gibi bir interface değil bir classtır. Context sınıfında bir c ve DbSet ile Entity türünde bir object türetir. Daha sonra bunları kullanarak ya tablolar üzerinde ekleme ve silme gibi işlemler yapar ya da bu tablolardan List gibi bilgi çeker. Bu generic bir yapıdır.



Aşağıda DataAccessLayer içindeki DataAccess class ını görüyorsunuz. Bu class executenonquery, dataset ve datareader gibi sık kullanılan değerleri döndüren methodlardan oluşuyor. Ayrıca connection durumunu kontrol etme işini de kolaylaştırıyor.

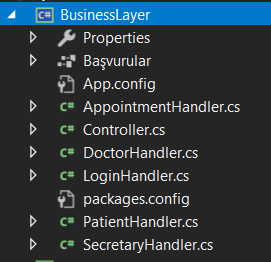


Aşağıda ise konsolda DataAccessLayer seçiliyken enable-migrations kodunu çalıştırarak otomatik olarak oluşturulan sınıfı görüyorsunuz. Buradan AutomaticMigrationsEnabled kısmını true yaptıktan sonra konsoldan update-database kodunu çalıştırarak oluşturduğumuz o tabloların nihayet veritabanına ilişkilendirilmesini sağlayabiliyoruz. Bu işlemleri yapmadan önce HMS deki app.config e connectionString bilgilerini ekledim.



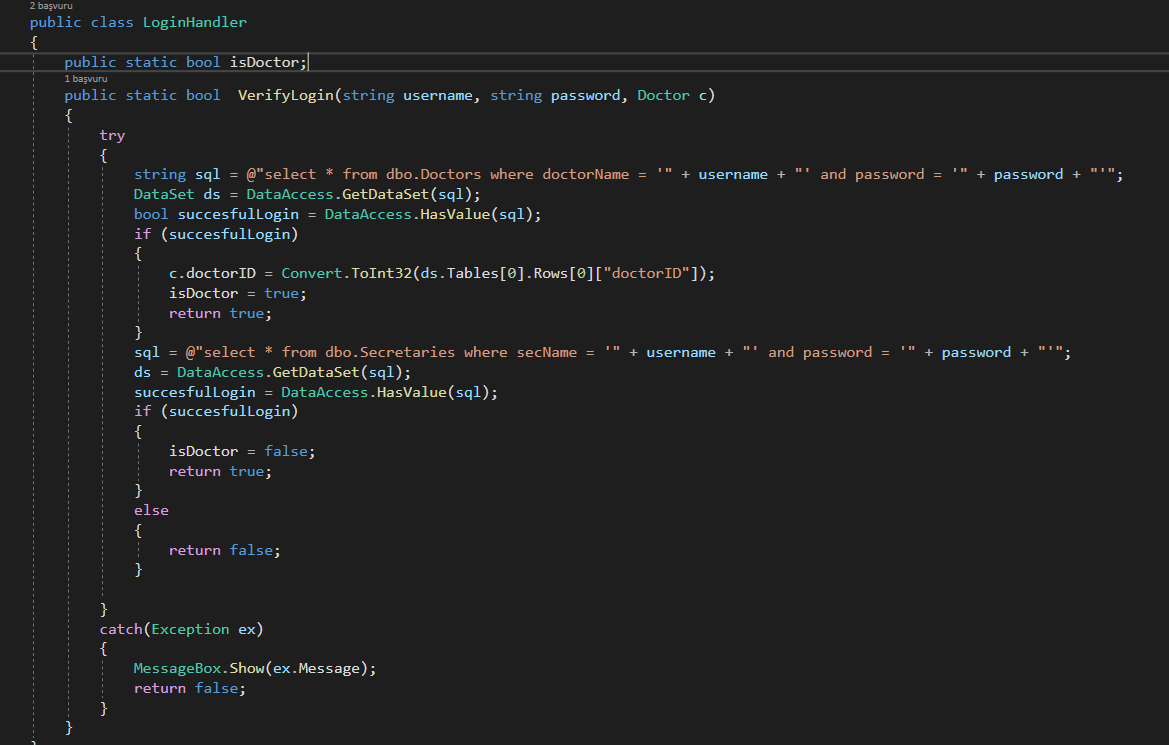
**BusinessLayer:**

Bu katman ise denetim katmanıdır. Burada veriler denetlenerek üzerinde işlem yapılıp yapılmayacağına karar verilir.



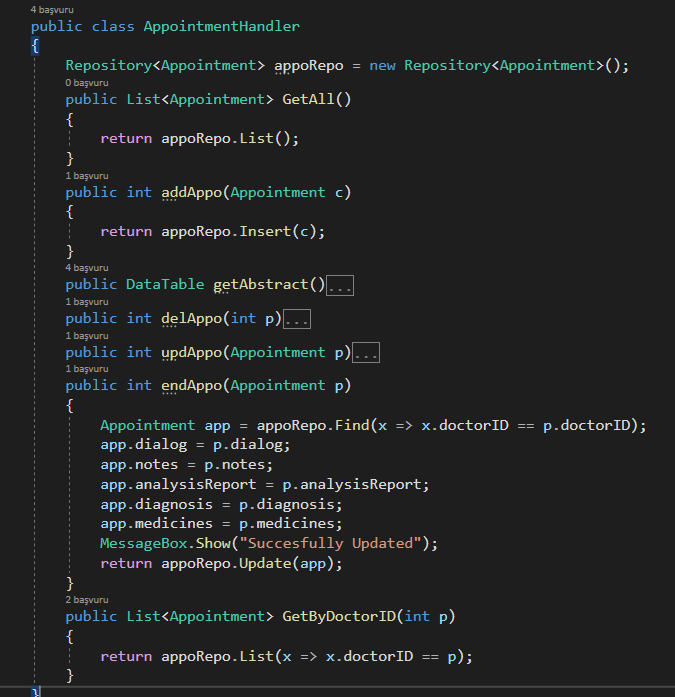
Aşağıda BusinessLayer daki LoginHandler sınıfını görüyorsunuz. Bu class önceden gösterdiğim Login sayfasını denetliyor. LoginForm unda girilen username ve password kodları alınıp bu kodların Doctors tablosuyla veya Secretaries tablosuyla eşleşip eşleşmediklerini denetliyor. Eğer giriş bilgileri iki tabloda da yoksa formda hatalı giriş yaptınız diye uyarı veriyor.

Giriş yaptığınız tablonun hangisi olduğuna göre LoginFormdan sonra hangi formun açılacağını belirlemek için isDoctor bool u kullanılıyor. Ayrıca eğer sisteme giriş yapan doktorsa doktorun id si de alınıyor. Daha önce formlar kısmında da bahsettiğim gibi bu id nin alınmasının sebebi sisteme giriş yapan doktorun kendisinden randevu alan hastaları görmesini sağlamasıdır.

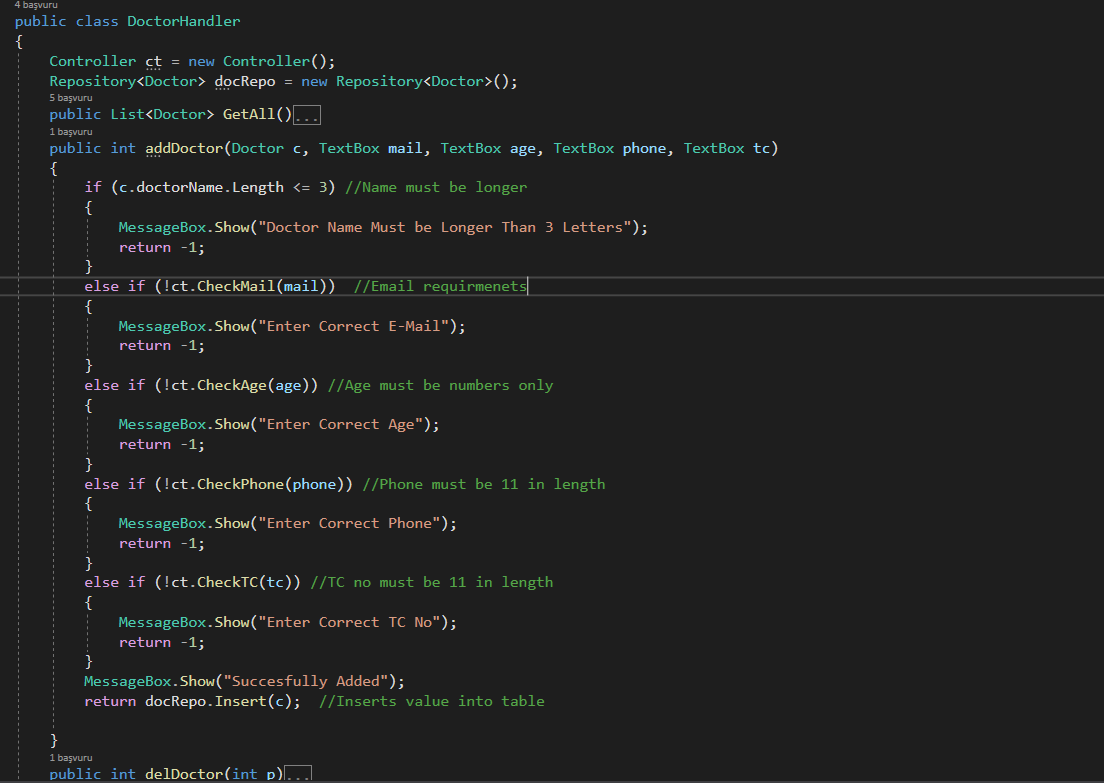


Aşağıda gördüğünüz ise AppointmentHandlerdır. Bu Appointment tablosunun denetleme işlemlerini yapıyor. List ve add metodları tabloları direk istenilen işlemleri yapıyor. Burada add metodunun denetlemesi yok ancak diğer tablolarda var. Mesela DoctorHandler kısmında ismin 3 harften daha uzun olması ve mail, telefon denetlemeleri gibi denetlemeler var. Silme ve Güncelleme kısımlarında ise önce find ile hangi kısımda işlem yapılacağı belirlenip işlemler öyle yapılır.

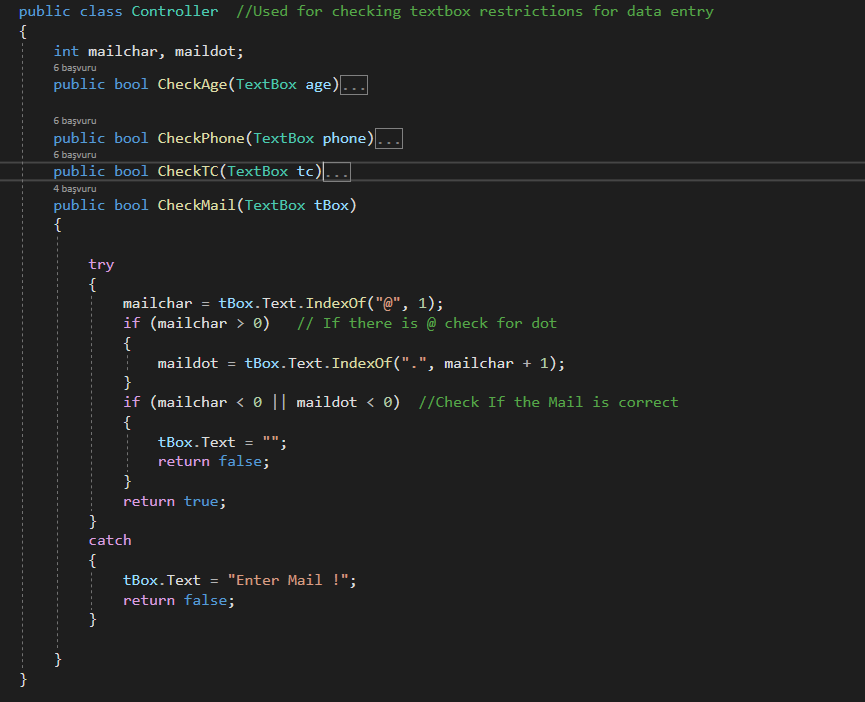
Appointment ilk önce sekreter tarafından tarafından doktor ve hasta eşleşmesi olarak yapıldıktan sonra doktor tarafından diğer bilgiler girilerek sonlandırılıyor. endAppo doktor tarafından sonlandırılan kısımdır. getAbstract ise sekreterlerin dialog ve notes gibi kısımları tabloda boş olarak görmemeleri için Appointment tablosunun sekreterlere sadece sekreterler tarafından görülmesi gerektiği kadarını göstermek için kullanılıyor.



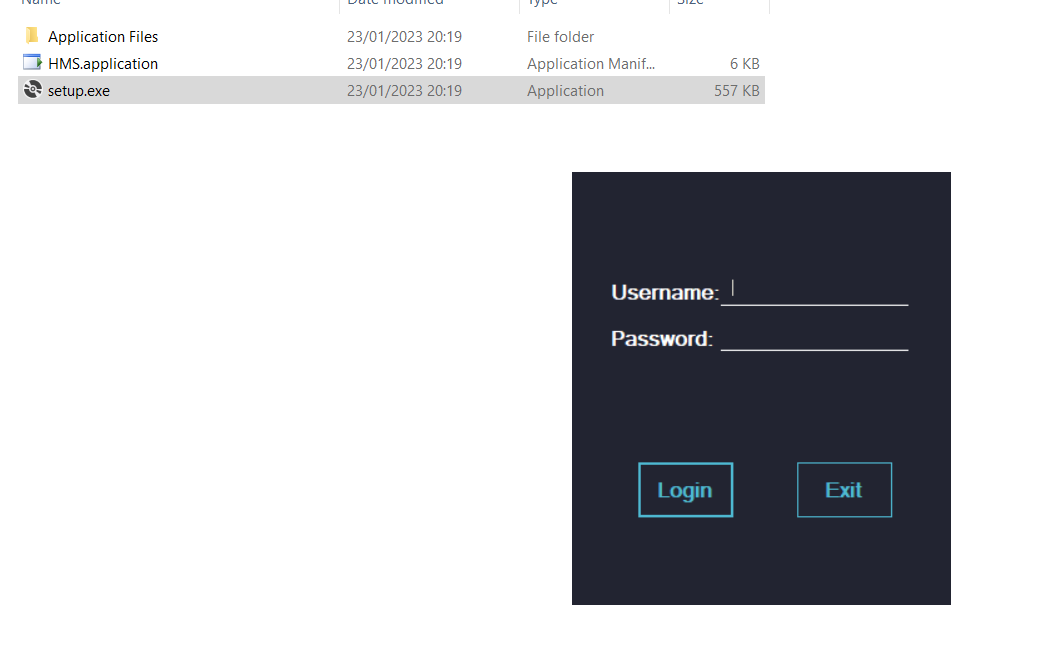
Önceden bahsettiğim gibi Doctors tablosuna ekleme yapmadan önce kullanılan denetlemeler. Tablonun geri kalanı silme, güncelleme gibi Appointmets tablosunda da olan metodlardan oluşuyor. Patients ve Secretaries Handlerları da DoktorHandler a büyük oranda benzemekte o yüzden sadece tekini raporda paylaşıyorum.

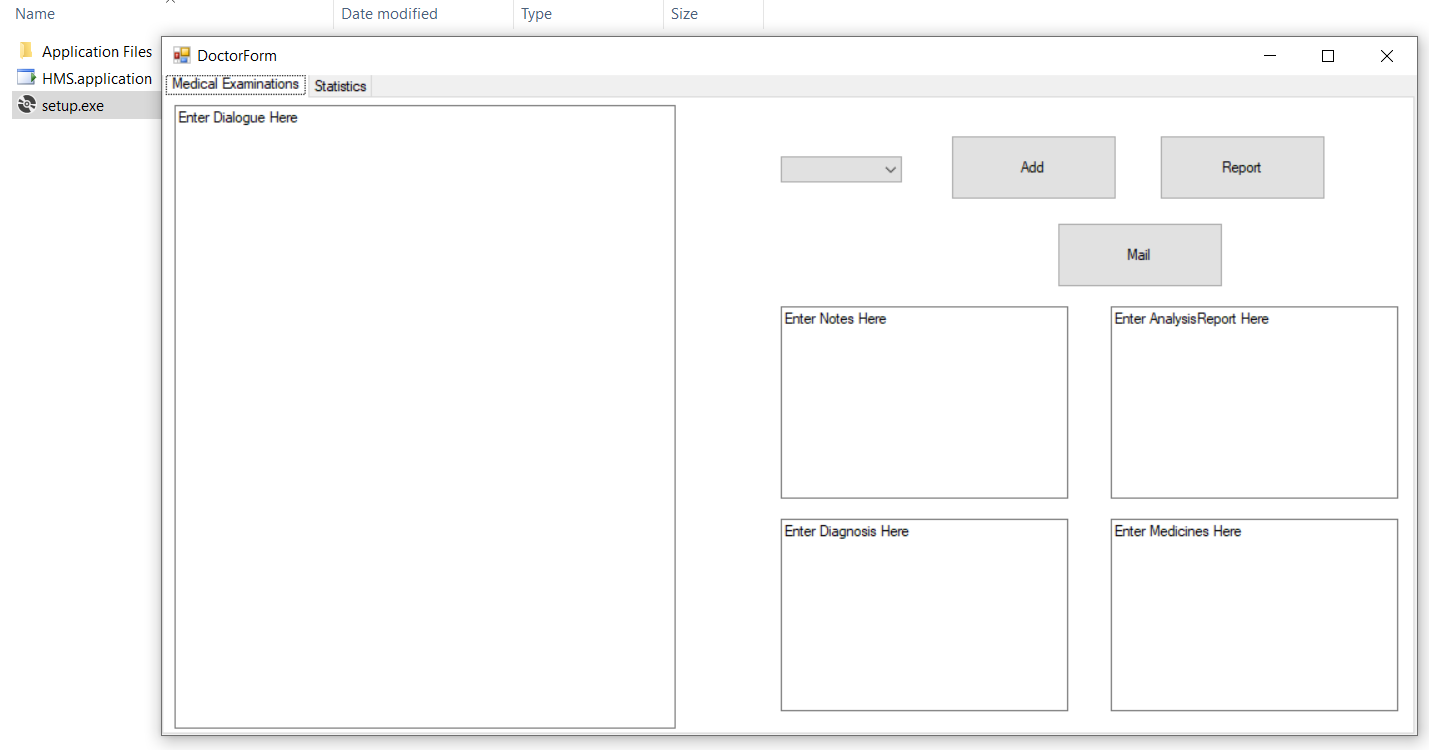
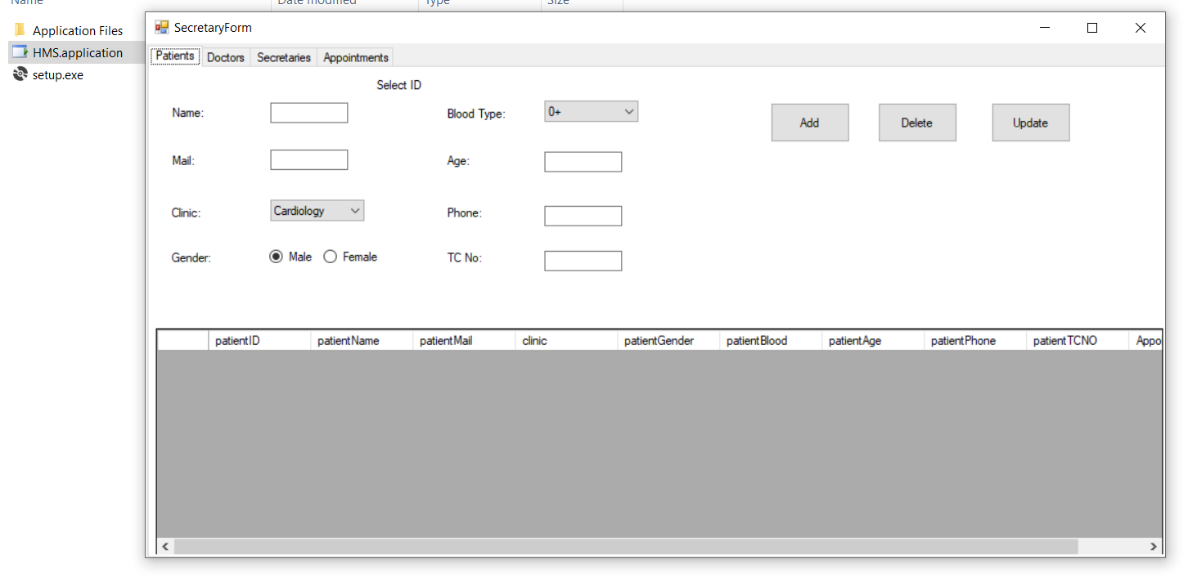


Aşağıda ise BusinessLayerda bahsedeceğim son class olan Controllerı görüyorsunuz. Burası DoctorHandler, SecretaryHandler ve PatientHandler classlarında kullanılan denetlemeler için. Girilen mailin içinde “@” ve “.” İşaretleri bulunmasını kontrol etmek telefon ve tcno nun sadece sayıdan oluşmaları ve 11 uzunluğunda olmalarını denetlemek gibi işlemleri yapıyor.



Aşağıda programın kurulduktan sonra çalıştırılmasının resimleri var.





**Mert Ali Uzun 181002007 İkinci Öğretim**